NN-Noxim: High-Level Cycle-Accurate NoC-based Neural Networks Simulator

Kun-Chih (Jimmy) Chen and Ting-Yi Wang
Department of Computer Science and Engineering, National Sun Yat-Sen University, Taiwan
kcchen@mail.cse.nsysu.edu.tw, tingyi@cereal.cse.nsysu.edu.tw

***Abstract*—Numerous advances have been made in developing intelligent system. The artificial neural network (ANN) was proposed, inspired by biological neural network, to solve the problems of pattern recognition, prediction and control optimization. While ANN provides the most advanced and accurate algorithms for many AI applications, it will consume much computing time and computing power. Hence, it is necessary to design a hardware efficient ANN accelerator for the future complex computing operation. However, due to the intensive computation and communication among each ANN neuron, the interconnection between each ANN neuron become complicated as the ANN size is scaling up. On the other hand, the network-on-chip (NoC) interconnection is proven as an efficient way to solve the problems of complicated multicore interconnection. Therefore, the NoC-based ANN design is an attractive way for the ANN hardware accelerator design. To facilitate the NoC-based ANN evaluation in the system level, we present a cycle-accurate NoC-based neural network simulator, NN-Noxim, in this paper. The classification precision of the simulation output is verified by the commercial high-level neural network simulator. Consequently, the proposed simulator can be used for the NoC-based neural network design, neural network computing design, and other related researches in the future.**

Keywords—NoC-based neural network simulator, NN-Noxim

I. INTRODUCTION

The Artificial Neural Network (ANN) has been applied to various applications such as image processing, speech recognition, machine translation *etc.* in the recent years [1]. The contemporary ANN comprises tens of layers and millions of parameters, which takes serious time consumption using general traditional computer. In order to solve these problems, more and more software and hardware have been proposed. In terms of hardware, the arithmetic mode of ANN rises the design challenges on how the hardware substrate efficiently load these parameters and perform the large neuron computing (*i.e.*, multiply-accumulation). Besides, the intensive computation and communication among each ANN neuron leads to complicated interconnection between each ANN neuron and worsens the difficulty to implement the ANN hardware accelerator [1].

To reduce the complexity of the interconnection among each ANN neuron, Kwon *et al.* proposed to adopt the network-on-chip (NoC) topology to interconnect each neuron [2]. The authors map each neuron of the involved neural network to the processing element (PE). Afterward, the data transaction between each neuron (*i.e.*, PE) will be performed through packet delivery. Because of the features of the scalable structure [2], the NoC-based neural network architecture is proper to realize the large-scale neural network on a single chip in the future, which receives much attention in the recent years [3][4][6]. To facilitate the NoC-based neural network design, a high-level NoC-based neural network simulators, which supports the analysis of power, latency, classification precision, is necessary.

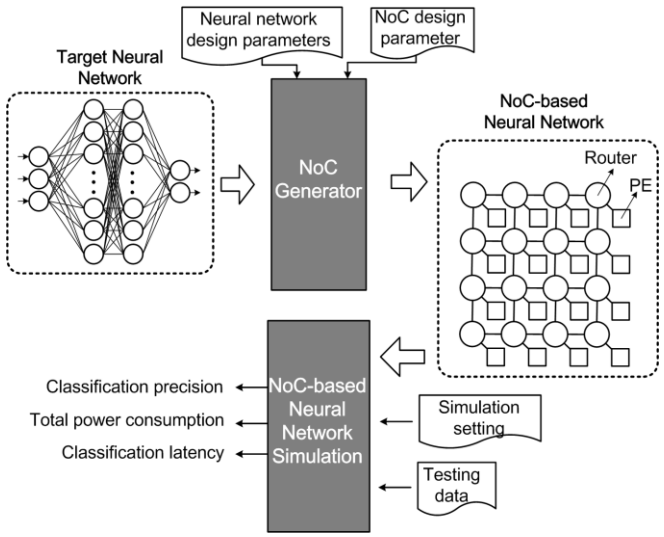

Fig. 1. The simulation flow of the proposed NoC-based neural network simulator.

In the recent years, many NoC system simulators have been proposed, such as Noxim [5], ATLAS [9] , Booksim [10] *etc.*. However, most of the NoC simulators focus on system performance under different traffic load and packet routing algorithms. Therefore, the current NoC simulators cannot simulate neural network operations. On the other hand, there are some mathematical software like NNtool in MATLAB [12] and GAN Playground [1] that can simulate the ANN computing under hardware settings such as fixpoint data type[7] or dynamically closes the operation of neurons[8]. However, these current neural network simulators still cannot support the aforementioned hardware matric analysis.

Because there is no proper neural network simulator can simulate the NoC-based neural network, we present a non-proprietary high-level cycle-accurate NoC-based neural network simulator in this paper and the simulation flow is shown in Fig.1. We extend a widespread NoC simulation tool, Noxim [5], and map the ANN neuron to the processing element (PE) of the involved NoC interconnection based on the NoC design parameters (*e.g.*, NoC size, routing algorithm,

978-1-5386-8552-5/18/$31.00 ©2018 IEEE

mapping algorithm, *etc.*). Afterward, the proposed simulator will simulate the generated high-level NoC-based neural network by using the input testing data. At last, the information of the classification precision, power consumption, and the classification latency can be analyzed after the simulation. The contributions of this paper are summarized as follows

1) We develop a high-level NoC-based neural network simulation platform and validate it to the commercial tool.
2) We extend a SystemC-based cycle-accurate NoC simulator to support the NoC-based neural network function, which can provide the results of hardware evaluation (*i.e.*, power, latency, classification precision *etc.*).

II. Related Work

A. NoC-based Hardware Simulator

A.1. Noxim [5]

Noxim is an open-source NoC simulator, which provides high flexibility to specify many different properties of NoC architecture such as network size, routing algorithm, *etc.*[5]. Based on the predefined NoC design parameters, the Noxim will create the user-defined NoC structure and the PE module generates the corresponding network traffic dataflow. Based on the mesh-based NoC model, the Noxim can simulate the router to transmit the received packet to the adjacent routers and analyzes the received packet to calculate throughput, delay, and power consumption. Although the Noxim can simulate the traffic behavior on the NoC platform, it only evaluate the NoC performance under the dataflow of the providing traffic pattern. Besides, the Noxim does not support any neuron computing function in each PE. Therefore, we cannot adopt it to simulate the ANN operation.

A.2. Booksim [10]

Booksim is a cycle-accurate interconnection network on chip simulator developed by the concurrent VLSI architecture group at Stanford University. Due to its modular architecture in C++, Booksim is very flexible. The current released BookSim 2.0 supports a wide range of topologies such as mesh, torus and flattened butterfly networks. Same as Noxim, Booksim provides diverse routing algorithms and other numerous options for customizing the router microarchitecture of the network. Although the functionality of Booksim has been continuously extended, Booksim still cannot support any process for ANN operation.

A.3. ATLAS [9]

ATLAS, which was built in java, is a 2D and 3D device simulator that performs DC, AC, and transient analysis for silicon, binary, ternary, and quaternary material-based devices. ATLAS enables the characterization and optimization of semiconductor devices for a wide range of technologies, and offers a variety of modules to simulate common behaviors in multi-core systems such as models for the reactor coolant system. Although ATLAS is a very power simulator to simulate many analogy results, it was develop only to analyze NoC traffic or overall computing performance under certain situation. Therefore, the ATLAS still cannot support any neural network function.

B. Neural Networks Simulator

B.1. Deep Playground [14]

To facilitate the neural network simulation, the Deep Playground, a kind of online neural network simulators, are released in recent years. The users can simulate a neural network computing directly based on the Tensorflow-based neural network frame. Besides, the Deep Playground can visualize training and computing processes. However, the Deep Playground only provides fully connected neural network with some common parameters in a very limited scale. Therefore, the flexibility of this tool is not enough. Besides, the Deep Playground cannot provide the hardware analysis.

B.2. GAN Playground [15]

GAN Playground is another online neural network simulators for Generative Adversarial Network(GAN). Unlike Deep Playground mentioned before, GAN Playground provide more parameters for user to adjust, and visualize more detail about process and result. Although GAN Playground is more flexible than Deep Playground, it provides insufficient database and scale which makes it still not a powerful simulator. In addition, the GAN Playground is unable to provide the hardware analysis either.

B.3. Neural Network Toolbox (NNtool) in MATLAB [12]

NNtool provides algorithms, pretrained models, and applications to create, train, and simulate both shallow and deep neural networks. Besides, the NNtool can provide the detailed analysis results based on the user defined parameter setting, such as neural network model, activation function, *etc.*. However, the NNtool cannot provide any information of hardware operation. Besides, the NNtool does not support the NoC-based ANN model.

III. Proposed NoC-base Neural Network Simulator

As mentioned before, although the Noxim is a widespread NoC simulator, it cannot support the neuron computing function in each PE, which cannot simulate the ANN operation. Therefore, we extend the Noxim and modify the transaction-level model (TLM) of Noxim in Fig. 2 to make it become a Neural Network Noxim (NN-Noxim), which can simulate the complete NoC-based ANN computing behavior. Fig. 3 shows the detailed NN-Noxim simulation flow. To facilitate the NoC-based ANN computing operation, the NN-Noxim adopts the following two phases:

1) *Neural network reshape*: Because it is hardware efficient to make a PE to operate the computing of multiple neuron, it is necessary to group the neurons, which will be mapped to the same PE. Therefore, it is necessary to reshape the adopted neural network for the further neuron mapping.
2) *PE neuron computing flow control*: After mapping the adopted neural network to the involved NoC platform, each PE will perform neuron computing while it receives all input data. Therefore, it is necessary to design a data flow control to prevent the invalid data flow (*i.e.*, the computing uncompleted data is delivered out to the next NoC node.)

The detailed descriptions of the two phases will be introduced in this section.

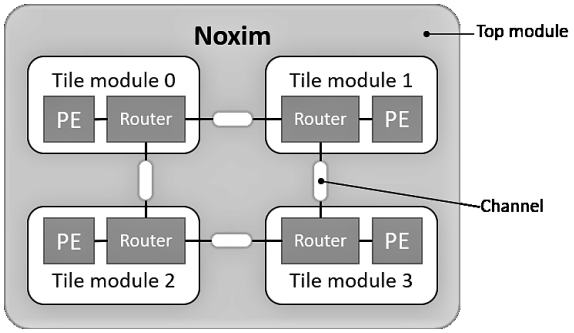

Fig. 2. The Noxim transaction-level model (TLM) overview.

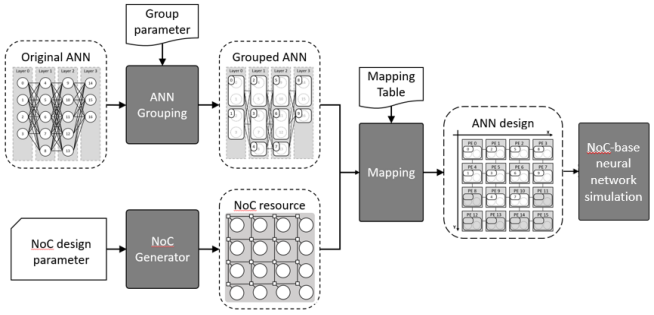

Fig. 3. The detialed simulation flow of the NN-Noxim to simulatethe NoC-bases ANN operation.

A. Neural Network Reshape

In order to achieve highly flexible data transmission, the data will be packetized and delivered by packet transmission. If each PE only compute one neuron process, the traffic load on the NoC system will become very heavy in the large scale ANN, which degrades the system performance significantly. Therefore, in the practical way, each PE will be assigned multiple neuron operations [3][4][6]. Hence, the target ANN should be reshaped. The each neuron of the reshaped ANN contains at least one neuron of the originally adopted ANN and we call the neuron of the reshaped ANN as the neuron group. The group neuron size depends on the processing limit of a PE.

Obviously, it is crucial that which neurons should be in the same neuron group. If the neurons in the same neuron group are not in the same layer of the target ANN (*i.e.*, the ANN before reshaping), the data transaction will become complicated, which results in heavy traffic load. To reduce the traffic load in the NoC after mapping the reshaped ANN, the neurons in the same neuron group should be in the same target ANN layer. The flow chart of the neuron grouping is shown in Fig. 4 and the Fig. 5 illustrates an example. In this example, we target a 4-layer ANN and the numbers of the neuron in each layer are 4, 5, 5, and 3 respectively. Besides, we assume the maximum processing limit of a PE is two neurons. By following the proposed neuron grouping method, we can reshape the target ANN in Fig. 5(a) to the reshaped ANN, as shown in Fig. 5(b).

B. Process Element flow control

As mentioned before, each PE of the involved NoC platform computes multiple neuron process. Besides, the

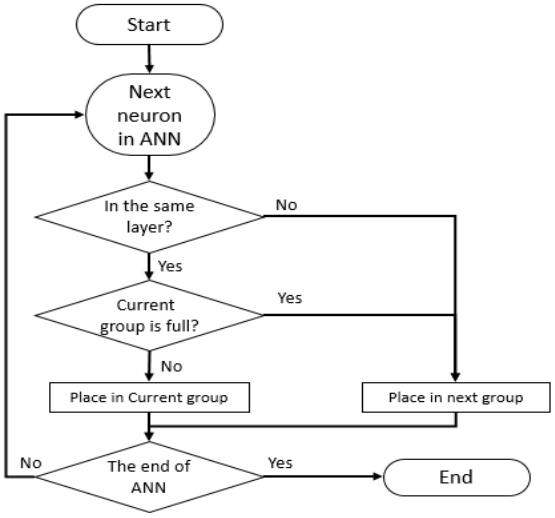

Fig. 4. Flow chart of group reshape.

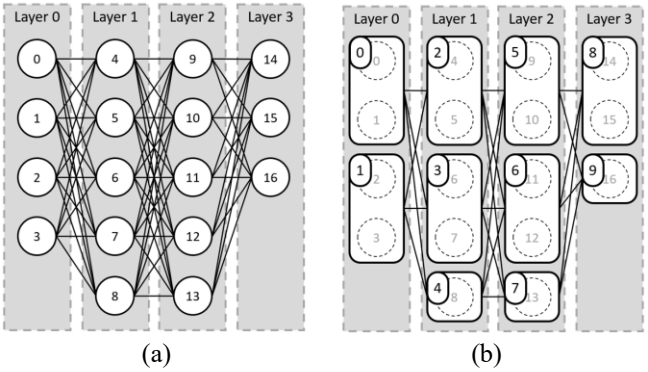

Fig. 5. (a) The original ANN model that was entered. (b) The ANN model after grouping

neurons, mapping in the same PE, are in an identical target ANN layer. Therefore, the PE will start the neuron computing process as long as receiving all the required data from the neurons in the last reshaped ANN layer. The flow chart of one PE is shown in Fig. 6. If there are *N* neurons mapping in the same PE, the PE will complete all the neuron computing as long as delivering out the output data of each neuron in the same neuron group (*i.e.* PE).

To facilitate the neuron computing, some weight tables should be adopted in each PE. Each weight table stores the weight, which should be involved for the specific neuron computing. When the PE receives the required data for the specific neuron computing, it will perform the multiply-accumulation process by using the corresponding weights stored in the corresponding weight table. The weight stored in the weight table can be obtained in the off-line training phase. At last, the result of one neuron computing is obtained after the non-linear activation function process. In this paper, the NN-Noxim supports four widely used activation functions, such as *ReLU*, *Sigmoid*, *Softmax*, and *Tanh*.

After obtaining the neuron computing results, the PE will packetize the data and deliver to the next PE for the further

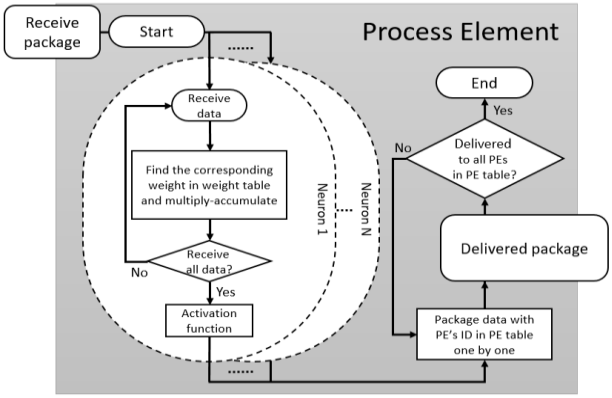

Fig. 6. The detial flow chart of the PEs in NN-Noxim.

neuron computing or final output. To identify the destination of the packet, each PE has its own destination table. Because of the forward process in the inference phase of the ANN computing, the destination of the data from each PE is fixed. Therefore, the destination table can be obtained after mapping the reshaped ANN to the NoC platform. By this way, we can ensure the dataflow during the whole ANN computing.

IV. Experimental Resulta And Analysis

A. Accuracy Verification of NN-Noxim

To validate the proposed NN-Noxim, we evaluate a fully connected neural networks in different size of hidden layer (*i.e.*, 1 to 4 layers). According to each hidden layer, there are eight neurons in each layer and the *Sigmoid* activation function is adopted in each hidden layer neuron. In addition, the *Softmax* is adopted as the activation function for the neurons in the output layer.

To compare the classification results between the proposed NN-Noxim and the MALAB neural network simulation tool, we involve the test data providing from the MATLAB NNtool, which is used to analyze the gender of crabs by their species or the size of various parts of their body. TABLE I. shows that the proposed NN-Noxim can achieve identical classification precision resulted by the MATLAB NNtool. Therefore, our proposed NN-Noxim can ensure the classification precision. To analyze the tradeoff between the classification latency and the classification precision under the NoC-based neural network computing, Fig. 7 shows that the classification latency is increased with respect to the classification precision.

B. Effectiveness of Group Size and Mapping

To analyze the performance under different group size and mapping algorithm, we adopt a 4-layer ANN to recognize the ten handwriting patterns (*i.e.*, zero to nine) in the MNIST dataset [11]. The numbers of the neuron of the target ANN are 784, 300, 100, and 10. Because the input data is a 28-by-28 grayscale two-dimensional image, the input layer of the target ANN should contain 784 neurons. Besides, the target ANN involves two hidden layer with widths of 300 and 100. At last, there are 10 neurons to output the confidence level of each digits (*i.e.*, zero to nine) provided by the *Softmax* activation function. On the other hand, to simplify the *NP*-hard mapping

TABLE I. The Classification Precision Validation

Size of the hidden layer	Classification Precision	
	MATLAB NNtool	***NN-Noxim***
8x4	100.00%	100.00%
8x3	93.33%	93.33%
8x2	93.33%	93.33%
8x1	90.00%	90.00%

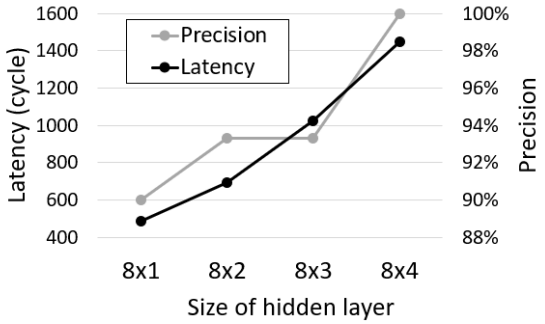

Fig. 7. The classification latency analysis under different size of the hidden layer.

problem [6], we implement five kinds of mapping algorithms in the proposed NN-Noxim:

1) *Dir_X*: The first type is direct mapping, which will directly and continuously map the neuron group to the PE according to the group ID. The *Dir_X* maps the neuron group to the PEs along X axis. Fig. 8(a) is the mapping result based on the reshaped ANN in Fig. 5(b).
2) *Dir_Y*: The *Dir_Y* maps the neuron group to the PEs along Y axis. Fig. 8(b) is the mapping result based on the reshaped ANN in Fig. 5(b). Therefore, accorded to an *N*-by-*N* NoC, the only condition of the two kinds of direct mapping algorithms is that total number of PE must be larger than the required neuron group number.
3) *Lyr_X*: The third providing mapping type is layer-based direct mapping, which will directly map the neuron group to the PEs according to the layer number in which the neuron group is located. The *Lyr_X* maps each reshaped ANN layer to the PE according to the X axis. Fig. 8(c) is the mapping result by mapping the reshaped ANN in Fig. 5(b) to a 4-by-4 NoC platform.
4) *Lyr_Y*: Different from the *Lyr_X*, the *Lyr_Y* maps each reshaped ANN layer to the PE according to the Y axis. Fig.8(d) is the mapping result by mapping the reshaped ANN in Fig. 5(b) to a 4-by-4 NoC platform. Therefore, accorded to an *N*-by-*N* NoC, the condition of these two layer-based mapping algorithms is that the *N* must be bigger than the maximum neuron group number of the reshaped ANN layer.
5) *NN_aware* [6]: In addition to the above four deterministic mapping algorithms, the user can also perform the developed mapping algorithm by inputting the designed mapping table. To demonstrate a customized mapping method, we implement the NN-aware mapping, which aims to find an optimal mapping results to minimize the data transactions during the NoC-based ANN operation.

TABLE II shows the performance evaluation by using identical target ANN under different mapping algorithms and

TABLE II. THE PACKDGE AND CYCLE ANALYSIS UNDER DIFFERENT GROUP SIZE AND MAPPING ALGORITHM

Group	Topology (after grouping)	Number of Packet/Flit	Mapping Algorithm (unit: cycles)				
			Dir_X	Dir_Y	Lyr_X	Lyr_Y	NN_aware [6]
32	25-10-4-1	294/9728	2508	3590	2580	3344	1916
64	13-5-2-1	77/4774	1782	2512	2022	2184	1412
128	7-3-1-1	25/2802	1600	1656	2006	1876	1226
256	4-2-1-1	11/1990	2004	2002	1214	1472	1214
512	2-1-1-1	4/1192	1204	1204	1204	1204	1204
1024	1-1-1-1	3/1190	1202	1202	1202	1202	1202

Note: Topology of the adopted ANN is 784-300-100-10 (before neuron grouping)

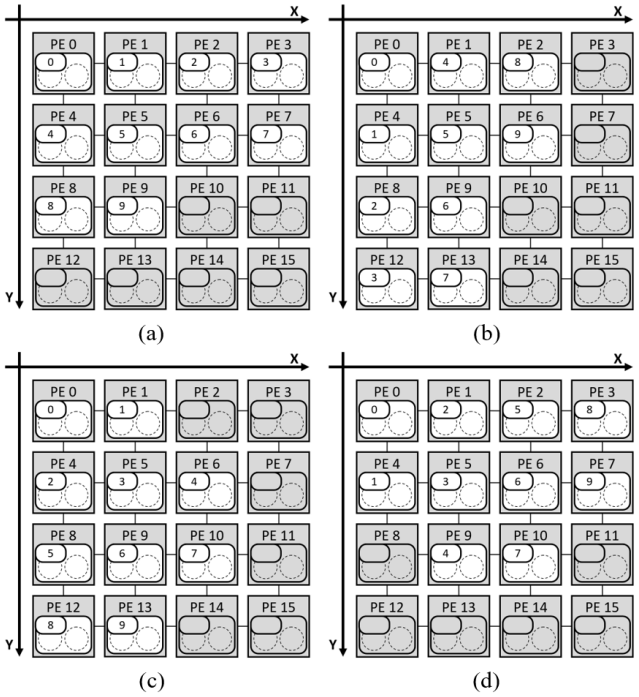

Fig. 8. (a)Group to PE direct mapping in X direction, and (b)in Y direction after grouping example show in Fig. 5. (c) Group to PE layer-based direct mapping in X direction, and (d) in Y direction after grouping example show in Fig. 5.

different neuron group size for the ANN reshaping. The XY routing is adopted in this experiment. Obviously, the larger neuron group size leads to better performance. The reason is that each PE can process more neuron computing, which reduces the data transaction traffic load (*i.e.*, the numbers of packets or flits during the NoC-based ANN operation) in the network. According to the adopted mapping algorithm, we can find that the mapping quality affects the performance significantly. Therefore, mapping algorithm design is also a crucial problem in the NoC-based ANN design.

V. CONCLUSION

In this paper, we propose a NoC-based neural network simulator. The validation shows our simulation results achieve the equal classification precision of the MATLAB. Besides, the tradeoff between the classification precision and the classification latency is analyzed. Besides, the performance is evaluated under different mapping algorithms. Hence, the proposed simulator can be used for the NoC-based neural network design, neural network computing design, and other related researches in the future.

ACKNOWLEDGMENT

This work was supported by the Ministry of Science and Technology, TAIWAN, under Grant MOST 107-2218-E-110-010, MOST 107-2218-E-100-006 and MOST 107-2218-E-100-004.

REFERENCES

[1] V. Sze *et al*., "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proc. IEEE*, vol. 105, no. 12, Dec. 2017.

[2] H. Kwon *et al*., "Rethinking the NoCs for Spatial Neural Network Accelerators," *Proc. ACM/IEEE int. Sym. Networks-on-Chip (NOCS)*, Oct. 2017, 8 pages.

[3] P. C. Holanda *et al*., "DHyANA: A NoC-based Neural Network Hardware Architecture" *IEEE Conf. Electronics Circuits and System*, Feb. 2016, pp. 177-180.

[4] J. Liu *et al.*, "Scalable Networks-on-Chip Interconnected Architecture for Astrocyte-Neuron Networks," *IEEE Trans. Circuits and System.s I: Regular Papers*, vol. 63, no. 12, pp. 2290-2303, Dec. 2016.

[5] V. Cataniz *et al.*, "Cycle-Accurate Network on Chip Simulation with Noxim," *ACM Trans. Modeling and Computer Simulation*, vol. 27, no. 1, article 4, Aug. 2016.

[6] X. Liu *et al.*, "Neu-NoC: A high-efficient interconnection network for accelerated neuromorphic systems," *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 141-146*,* 2018.

[7] J.L. Holi *et al.*, "Finite precision error analysis of neural network hardware implementations," *IEEE Transactions on Computers*, vol. 1, no 3, pp. 281-290, Jul. 1993.

[8] B. Reagen *et al*., "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," *Proc. ISCA, pp. 267-278, 2016.*

[9] A. Mello *et al.*, "ATLAS-an environment for NoC generation and evaluation," *Design Automation and Test in Europe (DATE)*. 2011.

[10] N. Jiang *et al.*, "A detailed and flexible cycle-accurate network-on-chip simulator," *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 86-96*,* Apr. 2013.

[11] L. Yann *et al*., "The MNIST database of handwritten digits," *URL http://yhann. lecun. com/exdb/mnist* , 1998.

[12] N. Wang *et al.*, *"*Application of Matlab/NNTool in Neural Network System [J]*," Computer Simulation,* vol. 4, pp. 125-128, 2004.

[13] M. Abadi *et al.*, "Tensorflow: a system for large-scale machine learning," *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI),* vol. 16, pp. 265-283, Nov. 2016.

[14] D. Smilkov and S. Carter. "A Neural Network Playground - TensorFlow," *URL https://playground.tensorflow.org*, 2017.

[15] R. Nakano. "GAN Playground - Experiment with GAN in your browser," *URL https://github.com/reiinakano/gan-playground*, 2017.